Binary Tree-Based Accuracy-Keeping Clustering Using CDA for Very Fast Japanese Character Recognition

Yohei Sobu
Graduate School of Information Sciences
Tohoku University, Japan

Hideaki Goto
Cyberscience Center
Tohoku University, Japan

Abstract

Real-time character recognition in video frames has been attracting great attention from developers since scene text recognition was recognized as a new field of Optical Character Recognition (OCR) applications. There are thousands of characters in some oriental languages such as Japanese and Chinese, and the character recognition takes much longer time in general compared with European languages. Speed-up of character recognition is crucial to develop software for mobile devices such as Smart Phones. This paper proposes a binary tree-based clustering technique with CDA (Canonical Discriminant Analysis) that can keep the accuracy as quite high as possible. The experimental results show that the character recognition using the proposed clustering technique is 10.2 times faster than the full linear matching at mere 0.04% accuracy drop. When the proposed method is combined with the Sequential Similarity Detection Algorithm (SSDA), we can achieve 12.3 times faster character matching at exactly the same accuracy drop.

1 Introduction

Real-time character recognition in video frames has been attracting great attention from researchers and developers since scene text recognition was recognized as a new application field of Optical Character Recognition (OCR) [1, 2, 3]. Koga et al. proposed a camera-based Japanese OCR for mobile phones [1]. Some mobile phones sold in Japan today are OCR-capable, although the performance is quite limited. We can find some OCR applications for Smart Phones as well. However, the OCR functions on such devices are only for still images, and real-time character recognition is currently under intensive research and development. Iwamura et al. developed a real-time character recognition method for European alphabets using an affine-invariant pattern matching technique [2]. The method, however, cannot be directly applied to Japanese characters since many of them have multiple connected components.

There are thousands of characters in some oriental languages such as Japanese and Chinese, and the character recognition takes much longer time in general compared with European languages. Speed-up of character recognition is crucial to develop software for mobile devices such as Smart Phones and PDAs (Personal Digital Assistances) because the processor performance is quite limited on those devices. In addition, reduction of computational complexity of character recognition is also very important because lower complexity leads to longer battery duration, which is one of the key factors in commercial success.

Faster character recognition is also beneficial for improving the character segmentation accuracy [1]. Japanese text is written or typed without a word space, unlike European languages, and this typesetting style makes the character segmentation difficult. Multiple hypothesis testing on the segmentation points is known to be quite useful for improving the character segmentation. A large number of character candidate images need to be processed, and therefore, a faster algorithm is required.

This paper proposes a binary tree-based clustering technique that can keep the accuracy as quite high as possible. CDA (Canonical Discriminant Analysis) is employed in the proposed method in order to achieve both higher accuracy and faster character matching than our previous method using PCA (Principal Component Analysis) [4]. In Sect.2, the basic, popular method of Japanese recognition is described. Some conventional speed-up techniques are introduced as well. In Sect.3, the new clustering technique and the algorithm for constructing the character dictionary are proposed. The fast character recognition algorithm is also given. Sect.4 gives experimental results and discussions.

2 Japanese Character Recognition and Speed-Up Techniques

2.1 Feature Vector-Based Japanese Character Recognition

Feature vector-based character matching is widely used in Japanese character recognition. A text line is segmented and character candidate images are extracted first. Some preprocessings such as size normalization are applied to each character image, and a high-dimensional feature vector is extracted from the processed image. A lot of feature vector definitions have been proposed so far. Peripheral Local Moment (P-LM) feature proposed by Hori et al. is one of the most precise features for Japanese character recognition [5]. Each feature vector is represented by a 576-dimensional vector with floating-point values.

Let an N-dimensional feature vector be given by

$$\boldsymbol{v} = (v_1, v_2, \ldots, v_N)^{\mathrm{T}} \ . \tag{1}$$

In the simplest form of character recognition, a set of representative vectors is used as the Character Dictionary. Each representative vector corresponds to a pattern class or a character. The representative vector is often defined by the mean vector of the feature vectors made from multiple fonts. Let N_{class} and N_{char} be the number of all classes and the number of all individual characters, respectively. We assume $N_{class} = N_{char}$ in this paper, although $N_{class} \geq N_{char}$ is possible to make advanced dictionaries.

Let the kth representative vector in the dictionary be represented by

$$\boldsymbol{m}^k = (m_1^k, m_2^k, \ldots, m_N^k)^{\mathrm{T}} \ . \tag{2}$$

The similarity between the representative vector and the feature vector of a character image to be recognized can be evaluated by a distance measure such as Euclidean one. The lower the distance is, the higher the similarity is. The class of the top candidate can be found by

$$k_1 = \underset{1 \le k \le N_{class}}{\operatorname{argmin}} \ d(\boldsymbol{v}, \boldsymbol{m}^k) \ , \tag{3}$$

where $d(\ldots)$ represents a distance function. The rth candidate k_r can be found by searching for the representative vector that yields the rth shortest distance.

A popular way is to measure the Euclidean distances in the N-dimensional space. In order to achieve faster processing, we may use squared Euclidean distance

$$d(\boldsymbol{v}, \boldsymbol{m}^k) = \sum_{i=1}^{N} (v_i - m_i^k)^2 \tag{4}$$

instead. Note that the output data, $k_1, k_2, k_3, \ldots$, remain exactly the same.

A secure way for evaluating (3) is to use the matching based on the full linear search (or exhaustive search) algorithm. Since Japanese OCR engines need to handle more than 3,000 different characters, the full linear matching having $\mathrm{O}(N_{class} \cdot N)$ time complexity takes a long time.

2.2 Speed-Up Techniques

Several approaches have been proposed for speed-up of character recognition.

2.2.1 SSDA

Sequential Similarity Detection Algorithm (SSDA) [6] is effective especially when the dimensionality is very high. SSDA aims at reducing the computational cost in the distance calculation. SSDA does never introduce error, and the output data remain exactly the same.

The full linear search is used in the SSDA-based character matching. We use a distance accumulator d_a and a variable d_S for keeping the current shortest distance. At the beginning of the search, d_S is set to a sufficiently large value. For each class k, the following steps are executed.

1. Set $d_a = 0$ and $i = 1$.
2. Repeat $d_a \leftarrow d_a + (v_i - m_i^k)^2$ and $i \leftarrow i + 1$ until $i = N_{class}$ or $d_a \ge d_S$.
   (The second condition should be tested at every s iterations, not at every time.)
3. If $d_a >= d_S$ then go to Step 5.
   (i.e. abort the current distance calculation)
4. If $d_a < d_S$ then set $d_S = d_a$ and $k_1 = k$.
5. $k \leftarrow k + 1$

If s is too small, the additional condition testing leads to longer running times. Since the optimal value of s is dependent on the input data, the order of classes in the dictionary, and the computer hardware, we set $s = 8$ empirically in this paper.

2.2.2 Coarse-to-fine approach

Coarse-to-fine approach are also effective in speed-up of character recognition. The input character image is roughly classified at the coarse recognition stage, and a group of characters is found. Within the group, fine recognition using the original representative vectors is performed. A clustering technique such as k-means is used to find the groups of similar characters.

Suppose we have N_{class} original classes and N_r clusters, each of which contains some classes. If we assume that the original classes are distributed to the clusters uniformly and that no member overlap between clusters is allowed, the average number of the classes in a group is N_{class}/N_r. The expected running time is $\mathrm{O}(N_r) + \mathrm{O}(N_{class}/N_r)$. However, some accuracy drop is unavoidable as no cluster overlap is allowed.

2.2.3 Dimensionality reduction

It is known that the dimensionality of the raw feature vector is in general too high for precise character recognition. The dimensionality can be made smaller to some extent by a subspace method. The smaller dimensionality contributes directly to higher processing speed, although the degree of the speed improvement is low if the original dimensionality is not so high. If a proper dimensionality is chosen and a good subspace is found, the accuracy could also be improved.

Note that various dimensionality reduction techniques can be combined with our clustering technique described in the next section.

3 Binary Tree-Based Accuracy-Keeping Clustering / Recognition

3.1 Accuracy-Keeping Clustering

We propose a simple yet powerful clustering technique using a binary tree combined with CDA (Canonical Discriminant Analysis). The constructed binary tree will be used as a BST (Binary Search Tree) in the fast character recognition algorithm.

In CDA, the dimensionality of the feature vectors must be smaller than the number of classes. In order to allow small sub-classes during the clustering, we apply a dimensionality reduction technique before the clustering. This can be achieved by using PCA (Principal Component Analysis). When the dimension of the feature space is reduced to n by transform matrix P calculated by PCA, the kth representative vector in the dictionary is derived from (2) as

$$\boldsymbol{m}_r^k = \boldsymbol{P}\boldsymbol{m}^k = (m_1^k, m_2^k, \ldots, m_n^k)^{\mathrm{T}} \ . \tag{5}$$

Figure 1 shows the data structure of the character dictionary and how the clusters of representative vectors are created. At the beginning, all the representative vectors of characters in the dictionary are put into the initial cluster (1,1). In other words, the cluster (1,1) contains N_{char} characters.

CDA is applied to the cluster and the first canonical vector (axis) $\boldsymbol{v}_c$ is found. The projection of the character k onto the first canonical vector is

$$r_{k,c} = \boldsymbol{v}_c \cdot \boldsymbol{m}_r^k \ , \tag{6}$$

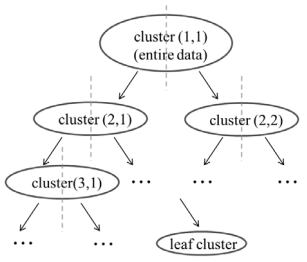

Figure 1. Binary tree-based clustering.

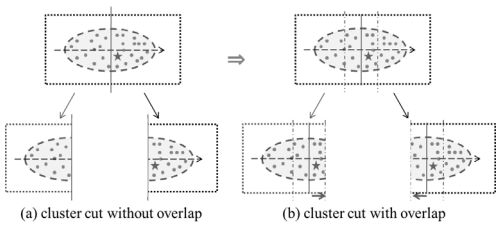

Figure 2. Cluster cut.

which is a scalar value, where the suffix c is the cluster identifier. The cluster is cut into two pieces at the cut point

$$H_c = \frac{1}{N_c} \sum_{k \in \text{cluster } c} r_{k,c} \ , \tag{7}$$

which represents the hyperplane in the n-dimensional space, where N_c is the cluster size. The clusters are recursively cut as shown in Figure 1 until one of the following conditions is satisfied.

$$K_1 \geq N_c \ , \tag{8}$$
$$K_2 \leq \max(l_{c,1}, l_{c,2}) \ , \tag{9}$$
$$K_3 \leq (\text{depth of the node}) \ , \tag{10}$$

where K_1, K_2 and K_3 are pre-defined parameters, and $l_{c,i} = N_{c,i}/N_c$ $(i = 1, 2)$ is the child cluster size divided by the current cluster size assuming the cluster is cut.

If we cut the cluster at H_c sharply without any overlap, the character recognition accuracy would deteriorate significantly as some classification error are introduced around the boundary. In order to keep the accuracy, we allow some overlap between the clusters as shown in Fig.2. The characters around the boarder H_c should belong to both clusters.

We take into account the variations in vector localization. Although the orientation of the first canonical vector is expected to be suitable for many characters, the other characters may have wider distributions in the feature vectors along the first canonical vector. In this method, the first canonical vector is found using the representative vectors. Then, for each font, the interval $r_{i,c} \pm \alpha \cdot \sigma_c$ is calculated, where α is a pre-defined parameter and σ_c is the standard deviation within the original class. If the cut point H_c falls in the interval, the corresponding character should belong to both clusters. If the different fonts for a character fall in both clusters, the character should belong to both clusters also (Fig.3).

3.2 Fast Character Recognition Algorithm

We construct a Binary Search Tree (BST) as shown in Fig.1 for fast character recognition. Each node has a pair of $\boldsymbol{v}_c^{\mathrm{T}} \boldsymbol{P}$ and H_c. Only the leaf node has the corresponding cluster data, which is the set of character identification numbers.

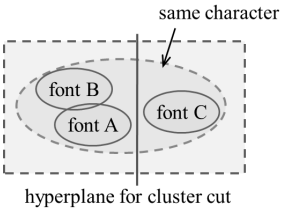

Figure 3. Cluster cut using multifont vectors.

Given an input character vector $\boldsymbol{v}$, the search begins at the root node. The product $(\boldsymbol{v}_c^{\mathrm{T}} \boldsymbol{P})\boldsymbol{v}$ is calculated and compared with the threshold H_c. When the search hits a leaf node of the BST, the character recognition by the linear search is performed within the cluster.

The time complexity is $\mathrm{O}((\text{tree height}) \cdot N) + \mathrm{O}(K_1 \cdot N) \simeq \mathrm{O}(K_1 \cdot N)$ if the tree height is small enough. Therefore, the expected overall speed-up is N_{char}/K_1 in our proposed method. In the extreme case where no cluster overlap is allowed and $K_1 \to 1$, the tree height becomes $\mathrm{O}(\log N_{char})$, which is much smaller than $\mathrm{O}(N_{char})$, according to the property of BST.

4 Experimental Results and Discussions

4.1 Experimental Setup

Throughout the experiments, we use 2,965 Kanji characters from JIS (Japanese Industrial Standards) Level 1. The following six TrueType font sets are used; IPA P Gothic, IPA UI Gothic, VL P Gothic, Sazanami Gothic, IPA P Mincho, and Sazanami Mincho. Each character image of size 56×56 pixels is rendered electronically. We use the leave-one-out method to evaluate the accuracies and the running times. In one configuration, one of the font sets is used as the unknown data, and the other five are used for dictionary. All the measured values are represented by the mean values of all six configurations unless otherwise mentioned.

The PC used in the experiments is equipped with Athlon 64 3700+ (2.4GHz) and 2GB memory.

We evaluate the performances of our PCA-based method [4] as well for comparison purposes. The clustering and matching algorithms of the former method are the same as those in our new method except that PCA is used instead of CDA and that no dimensionality reduction is used.

4.2 Parameter Tuning

For comparison purposes, we set $K_2 = 0.97$ and $K_3 = 17$, which are the same values respectively as those in [4].

The impact of K_1 to the running time is greater than that of α. We can easily see that a combination of small K_1 and large α yields shorter running time with respect to the same accuracy. $K_1 = 300$ is used in the rest of this paper. To make the feature dimensionality lower than $K_1 = 300$, PCA is applied to the dictionary in the original 576-dimensional feature vector space, and the top 256 eigenvectors are used as the basis vectors of the new 256-dimensional feature vector space.

Table 1. BST size ($K_1 = 300$, input font: IPA P Gothic).

	α	num. of clusters	ave. depth of clusters	ave. num. of chars./cluster
CDA	0.2	37	5.3	247.1
	0.4	118	6.9	237.0
	0.6	524	9.0	260.7
PCA[4]	0.6	4,492	12.3	269.3

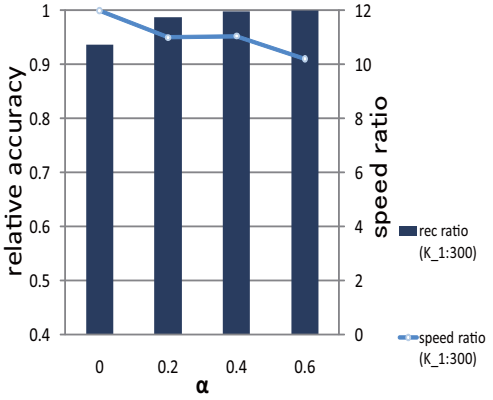

Figure 4. Relative accuracies and speed-up ratios.

4.3 Performance Evaluation

Figure 4 shows the relative accuracies and the character matching times with respect to various α values. The original accuracy obtained by the full linear matching is 99.75%, and the matching time is 17.0sec. The average time per character is 5.73ms, which corresponds to 5.82 characters per video frame at 30fps.

Higher α is desirable to keep the accuracy as high as possible, although the speed-up ratio deteriorates as the parameter α increases. We have achieved very high relative accuracy 99.96% and speed-up ratio 10.2 at $\alpha = 0.6$.

Table 1 shows the BST size. Compared with our PCA-based method [4], the number of clusters is much smaller in our new method. The amount of memory for the index table (cluster data) is 2.6MB in the proposed method, while the method in [4] requires 22.1MB. Thus, the new method is quite efficient in memory consumption.

4.4 Performance Comparison

Some conventional speed-up techniques can be used together with the proposed method as described in 2.2. By introducing SSDA, we can obtain some speed improvement without sacrificing the accuracy.

Table 2 shows the performance comparison using various speed-up methods. When SSDA is used together, the speed-up ratio reaches 12.3. The processing time 1.38sec corresponds to 71.6 characters per video frame at 30fps. Further speed improvement would be necessary when we develop a real-time on-the-fly OCR application for mobile devices, although the achieved speed is satisfactory on a desktop computer.

The proposed CDA-based method outperforms the PCA-based method [4] with respect to all the aspects of accuracy, speed, and memory consumption.

Table 2. Performance comparison using various speed-up methods.

	accuracy (%)	time (sec)	relative speed
full linear matching	99.75	17.01	1 (ref.)
full linear matching with SSDA	99.75	11.92	1.43
proposed method	99.71	1.67	10.2
proposed method with SSDA	99.71	1.38	12.3
PCA-based method [4] ($\alpha = 0.7$)	99.53	2.05	8.30
PCA-based method [4] with SSDA ($\alpha = 0.7$)	99.53	1.72	9.89

The higher accuracy is probably due to the fact that CDA takes into account the intra-class variances while PCA does not in finding the principal vectors.

5 Conclusions

Very fast Japanese character recognition algorithms are crucial for developing software for mobile devices, for realizing real-time OCR, and even for improving character segmentation performance. In this paper, we have proposed a binary tree-based clustering technique with CDA that can keep the character recognition accuracy quite high. The BST-based character recognition algorithm runs at the speed 10.2 times faster than the full linear matching with merely 0.04% accuracy drop.

Further optimization of the clustering, especially the cluster cut criteria, is included in our future work.

Acknowledgement

A part of this work was supported by Grants-in-Aid for Scientific Research No.22300194 from JSPS.

References

[1] M. Koga, R. Mine, T. Kameyama, T. Takahashi, M. Yamazaki, and T. Yamaguchi: "Camera-based Kanji OCR for Mobile-phones: Practical Issues," Proc. of ICDAR 2005, pp. 635–639, 2005.

[2] M. Iwamura, T. Tsuji, A. Horimatsu, and K. Kise: "Real-Time Camera-Based Recognition of Characters and Pictograms," Proc. of ICDAR 2009, pp. 76–80, 2009.

[3] H. Goto and M. Tanaka: "Text-Tracking Wearable Camera System for the Blind," Proc. of ICDAR 2009 Volume 1, pp. 141–145, 2009.

[4] Y. Sobu, H. Goto, and H. Aso "Binary Tree-Based Precision-Keeping Clustering for Very Fast Japanese Character Recognition," Proc. IVCNZ 2010.

[5] K. Hori, K. Nemoto, and A. Itoh: "A Study of Feature Extraction by Information on Outline of Handwritten Chinese Characters — Peripheral Local Outline Vector and Peripheral Local Moment —," Trans. IEICE D-II, vol. J82-D-II, no. 2, pp. 188–195, 1999 (in Japanese).

[6] D. Barnea and H. Silverman: "A Class of Algorithms for Fast Digital Image Registration," IEEE Trans. on Computer, vol. C-21, no. 2, pp. 179–186, 1972.